CHAPTER

9

# Active Directory Maintenance and Recovery

**After completing this chapter, you will be able to:**

- ♦ Describe how data is stored in Active Directory
- ♦ Define key Active Directory terms such as System State Data
- ♦ Perform a non-authoritative restore of Active Directory data
- ♦ Understand the security permissions required to perform a backup and restore

Active Directory is a key part of your Windows 2000 network. A lot of time is spent designing its architecture, both from a logical and physical standpoint. But when you get down to it, Active Directory is simply another set of files that exist on a server in your enterprise. As part of your plans, you should make sure that you have the ability to perform both a backup and a restore of Active Directory data.

In this chapter, you will learn about the data structure of Active Directory. Once you understand how data is stored and where the physical files are stored on the hard disk, we will show you how to perform system maintenance, backup, and restore operations.

## Recovery and Maintenance Overview

When things are working well, the network is humming, and your servers are operating without a single critical message in your event logs, it is easy to forget how difficult the situation can become in a relatively short period of time. A simple call to a help-desk technician saying, "You know, I can no longer save my documents" can quickly turn the life of a system administrator into a turbulent race against time.

When it comes to disaster recovery, it is not a matter of "if"—it is a matter of "when." I can guarantee that at some time in the future, you will be standing before a server that has either crashed outright or has ceased to perform its function sufficiently. If it happens to be a server that is home to a mission-critical piece of software (such as an e-mail server), people will be buzzing around you like flies, pleading with you to bring the server back online as soon as you can. You must understand what files and data need to be backed up—and you must execute that backup plan—or you might very well never be able to bring your server back online. We will cover everything you need to know to make sure that Active Directory operates on your network, even if one of your domain controllers goes belly up.

As you have worked through this book, you have no doubt noted the way Windows 2000 domain controllers (DCs) differ from those in previous versions of Microsoft Windows NT. One of the key differences (although not the only difference) between these two sets of DCs is that in Windows 2000, DCs are peers. That is, the whole concept of a Primary Domain Controller (PDC) and Backup Domain Controller (BDC) no longer exists. In the old world, if a BDC went down, it was not a disaster. If the PDC went down things could get pretty scary, but the situation was easy enough to fix.

You can be forgiven for thinking that because DCs in Windows 2000 are peers, it should not be quite so catastrophic if one of them dies. Actually, that may or may not be the case. Don't forget that Windows 2000 has a concept of **sites** (see Chapter 2), and sites affect much of the normal operations on your network. For instance, sites are used to optimize logon traffic on your network, to give access to the nearest Distributed file system (Dfs) share, and for Active Directory replication traffic, among other things. If a carefully placed DC goes down, the effects on your network can be unpredictable and obscure. Although this event would probably not prevent users from doing their work, the balance of your network can suffer considerably.

So, the same rules we have used for user data still apply. The only ways to protect yourself are with fault-tolerant hardware (such as Redundant Array of Independent Disks, or RAID 5) and careful backup and restore policies. That is one of the purposes of this chapter: to make sure that you have an understanding of what constitutes a reasonable backup of a DC in Windows 2000.

Before we begin looking at the mechanics involved, you need to have an understanding of how Windows 2000 stores data. By understanding this process, you will be able to see how Windows 2000 DCs are able to survive if a DC is suddenly turned off. We will then locate the critical Active Directory files and tell you how to back them up.

## ACTIVE DIRECTORY DATA MODEL

Active Directory is a lot of things, but if we were asked to come up with a two-word definition, it would be this simple: a database. (Given three words, we'd say it is a distributed database.) Along with databases such as Microsoft SQL Server, the underlying database of Active Directory has built-in mechanisms that ensure data is written in a uniform, guaranteed way. If this were not the case, then there would be a good chance that the data would become corrupt. Figure 9-1 shows the basic structure of the Active Directory data store model. As you can see, it has three distinct layers.

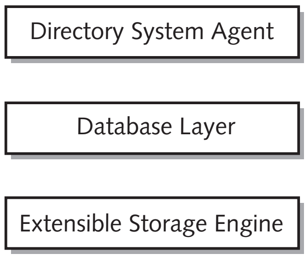


**Figure 9-1** Three layers of the Active Directory data store model

Let's take a look at each of these items in turn, and present a clear definition for each.

### Directory System Agent (DSA)

When you use an application to access data within Active Directory, the data must first go through the Directory System Agent (DSA). The DSA sits at the top of Figure 9-1, and it creates an instance of the directory service. When an application wants to access data within the directory, the DSA makes it available.

You should note that applications within Windows 2000 do not have direct access to the underlying data in the directory. Instead, the DSA impersonates applications, retrieves the data itself, and then passes it back to the calling application. This process offers an extra level of security. Because applications and users are not able to access the underlying data, the data does not have to be secured through countless access control lists.

The DSA has many tasks to perform. Because Active Directory is a fairly complex process, it is not enough for the DSA simply to make data available. It must also enforce security settings within the directory and facilitate Active Directory replication.

### The Database Layer

Some key concepts regarding the physical storage of Active Directory data might surprise you. When looking at Active Directory, and when studying topics such as the schemas and domain design documents, you are used to viewing the data in a hierarchical form—that is, like a tree with the trunk at the top and roots spreading out beneath it.

The physical data as it is stored in the database is not like that. In fact, the data within the physical file is **flat**. A hierarchical view is simply a representation of what actually exists.

The database layer takes that flat data and makes it hierarchical for you. It applies the defined schema for Active Directory against the data to show you what you expect to see. It manages all the parent-child relationships in your management tools and provides a better way of seeing your data. After all, looking at one huge table is certainly not conducive to easy learning!

### Extensible Storage Engine (ESE)

The Extensible Storage Engine (ESE) is the underlying engine that physically stores the Active Directory data. This engine scales to enormous proportions. We will be taking a closer look at how the integrity of the data is ensured in a moment.

Note

It might surprise you to learn that Active Directory data is stored in just two tables. The first table, which is known as the **object table**, holds the data. The other is the **link table**, which implements the links between objects that help build relationships.

It turns out that this underlying database engine is not new. In fact, it has been used in production networks for quite some time. It is the same engine that is used in Microsoft Exchange 5.5. Because it was thoroughly field-tested prior to the existence of Windows 2000, Microsoft is assured of performance and scalability issues. This database was designed to handle up to 16 terabytes of data—so you can be confident that it will scale across your enterprise.

## How Data Is Written to the Database

Now that you have a good idea what the structure of the underlying engine is like, it is time to walk through the process of how data is written to Active Directory. The method Active Directory uses is very much like that used in Microsoft SQL Server. Data is never written directly to the database; instead, it is held temporarily in a log before the physical write takes place. This log can then be used to fix problems, should they occur.

A change being made to the database is called a **transaction**. Transactions can come from many places: from the administrative tools provided with the operating system, from scripts written by users and administrators, or from third-party software vendors who wish to integrate their products with Active Directory. Regardless of where they come from, the process of moving the data from the interface to the physical data store (the object table in the ESE) is a regimented and efficient process.

The following paragraphs walk you through the process of how a change is made in a console and is eventually written to the object table. As you will see, it is a simple yet effective way of ensuring that data is written to the database efficiently, and with a level of fault-tolerance necessary in such a critical application. Figure 9-2 shows an administrator sitting at a console ready to make that change.

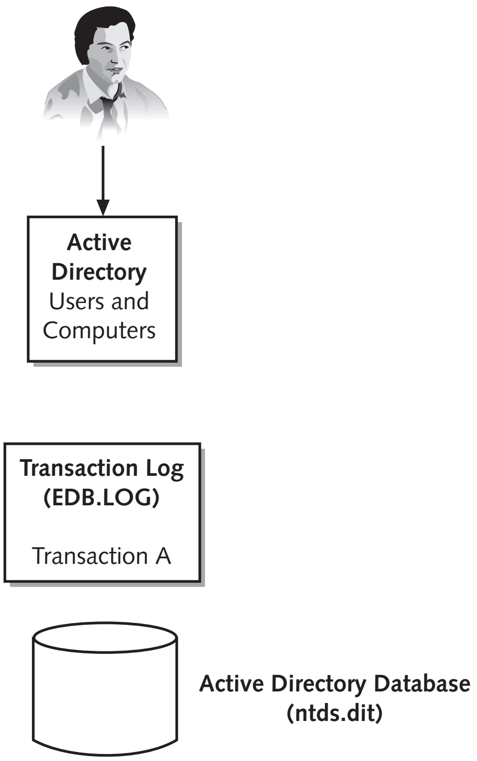

Active Directory Users and Computers
Transaction Log (EDB.LOG)
Transaction A
Active Directory Database (ntds.dit)

**Figure 9-2** An administrator making a change using an administration tool

After the administrator selects the necessary options, the change is made on the DC. First, the change is accepted and a transaction is created. This transaction encompasses the changes that will be made to the underlying database records. Next, this transaction is written to a **transaction log**. A transaction log is a file stored on disk that contains a list of all the transactions that have been applied to the Active Directory database. It is important to realize that all proposed changes to Active Directory are written to a transaction log *before* actually going to disk. Figure 9-3 shows this data being written to the transaction log.

In the case of a failure on a DC, Active Directory can look at the list of changes stored within this file and quickly determine which changes it has stored in the physical database and which changes have been lost due to the failure. It uses this information either to undo changes that have not been fully written to the database file (known as **rolling back** a transaction, because it undoes an incomplete change) or to apply all outstanding transactions (known as **fully committed** transactions).

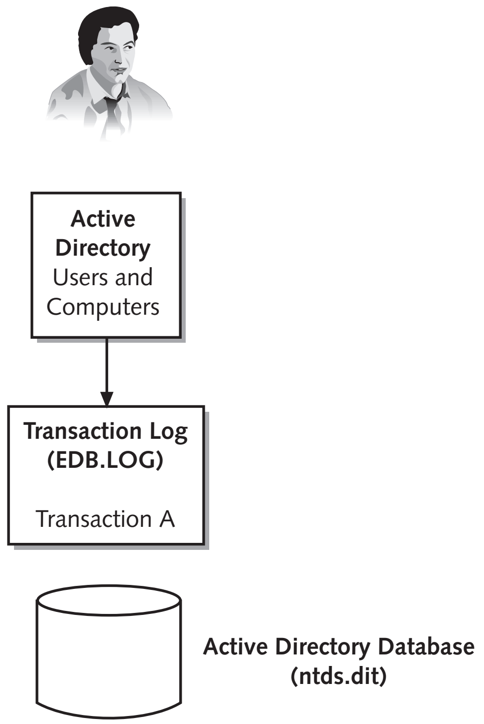

Active Directory Users and Computers
Transaction Log (EDB.LOG)
Transaction A
Active Directory Database (ntds.dit)

**Figure 9-3** Data is written to the transaction log

The next step is completed when the transaction is written to the database stored in memory on the DC. The entire database is not stored in memory (so you don't have to worry about the enormous amounts of Random Access Memory [RAM] your DC will need); instead, a small piece of the database is stored and written to memory. These pieces of the database are known as **pages**. (A discussion of pages is outside the scope of this book, and the topic is not important to pass the 70-217 certification exam. Readers are nevertheless encouraged to read about the topic for their general knowledge.)

Finally, the change is written to the physical file on the disk. This step is shown in Figure 9-4. The transaction can now be removed from the transaction log, because the data is no longer needed. The transaction log uses a pointer system to determine which entries have been physically written to the data store on disk and which are still waiting to be written. The most up-to-date view of Active Directory data is the physical data store, along with any entries in the transaction log that are waiting to be written to that physical store.

When is data written from the transaction log to the physical data store? The answer varies. The official answer is that the data is written during idle times on the DC. So, you cannot count on the preceding steps occurring at the same intervals for each transaction.

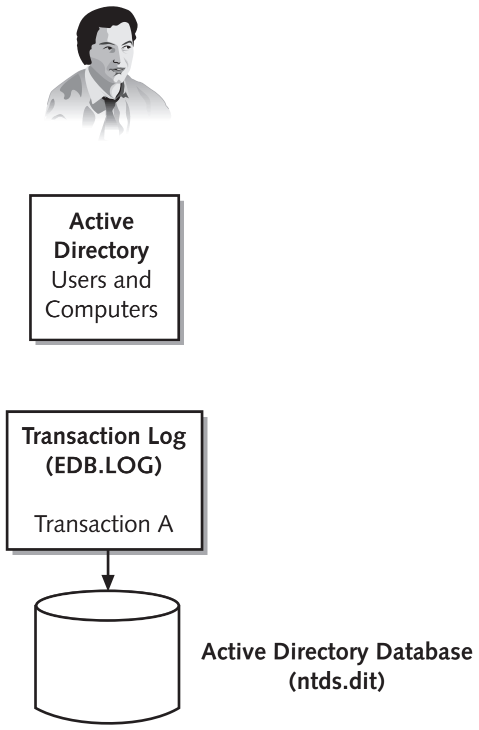

Active Directory Users and Computers
Transaction Log (EDB.LOG)
Transaction A
Active Directory Database (ntds.dit)

**Figure 9-4** Data is written to the database

Note: Have you noticed that after you make your computer a DC, it seems to perform more poorly than before? Not only does Active Directory place an additional burden on DCs by installing a host of new services and processes, it also disables write caching to the hard disks. It does so to ensure data integrity of the Active Directory data files. With write caching enabled, Windows 2000 cannot be sure precisely when data is written to the hard disk.

You might be thinking that once the data within a transaction log has been committed to the database, there is little point in keeping it around forever—and you would be right. A short archive period is probably a good idea, because you never know when a failure will take place; but storing information for days is not a good use of the finite resources on the DC (meaning the hard disk space). In fact, Windows 2000 periodically performs what is known as **garbage collection**. This process runs every 12 hours on DCs; it cleans up the transaction logs, getting rid of entries that are no longer of any use. (We will talk more about this process later in this chapter.)

## THE FILES OF ACTIVE DIRECTORY

Having taken a look at the process that enables data to be written to Active Directory, let's examine the actual files that are affected. The following sections give the location and names of both the database and transaction files that are used by Windows 2000. These are the files you should back up for disaster-recovery purposes.

### The Database File (ntds.dit)

The ntds.dit file contains both the object and link tables. This is obviously a critical file that must be saved for disaster-recovery purposes. ntds.dit can be found (by default) in the <*systemroot*>\NTDS folder on each DC. In addition to information on all the objects of the directory, this file stores the schema and configuration partitions of the directory.

If you do a search of your hard disk, you will turn up two copies of ntds.dit. In addition to the copy in the \NTDS folder, another version of the file is stored in <*systemroot*>\system32. This copy exists on all Windows 2000 servers and is used as a template, should the server ever be promoted to a DC. If this file is missing, then the Windows 2000 CD is needed during the promotion.

Note

The DIT file extension stands for Directory Information Table.

### The Log Files (EDB.LOG, RES1.LOG, and RES2.LOG)

The EDB.LOG file is the transaction log that we looked at earlier in this chapter. It is used to store transactions before they are committed to the database file (ntds.dit). After that process has taken place, these logs have a record of which transactions have occurred on a DC.

The log file has a predetermined maximum size. Once that size is reached, the file is renamed, and a new transaction log is created. The approximate maximum size of this log is 10MB. Older transaction log files are stored in the same folder as the EDB.LOG file, which by default is the same location as ntds.dit. The older logs are named EDB*****.LOG, where the asterisks are replaced by a hexadecimal value.

It can be difficult to calculate how quickly transaction logs will grow. Transactions are of differing sizes, depending on the amount and number of changes that need to be committed. So, it is virtually impossible to determine precisely when one log file will be considered full and another will be created.

As you have seen, the transaction logs play a vital role in the maintenance and integrity of Active Directory data. But what happens if the hard disk runs out of disk space, and there is no longer enough space in the current transaction log to record all the changes that need to be made to the directory? Two transaction logs are standing by to take over if such a disaster should occur: RES1.LOG and RES2.LOG. These files are 10MB each (not coincidentally, the default maximum size for the EDB.LOG file). In effect, they reserve space on the hard disk, should that space ever be needed by Active Directory.

#### Circular Logging

You may wonder how much disk space the transaction logs will take up on your DCs. We hope that each of your DCs has enough disk space to perform its intended role; but if you are particularly concerned about this topic, you might want to consider **circular logging**.

Circular logging is a method whereby transaction logs are not archived when they are full. Instead of having a slew of EDB*****.LOG files on your DC, the same file is used repeatedly—when the EDB.LOG file is full, it is overwritten. This mechanism can help you maximize the space on your DC's hard disks, but it offers less fault tolerance.

Caution

Unfortunately, turning on circular logging requires a Registry hack. I don't want to go through the usual warnings about editing the Registry—just keep in mind that if you mess up the Registry on a Windows 2000 machine, it's probably going to be toast! The Registry key for turning on circular logging can be found at HKEY_LOCAL_MACHINE\CurrentControlSet\Services\NTDS\Parameters\CircularLogging. A value of 0 means that it is turned off; changing the value to 1 turns the feature on.

### Checkpoint Files (EDB.CHK)

So, how does Active Directory recover when power is suddenly lost to a domain controller? As you have seen, Active Directory is maintained through the interaction of a database file and transaction logs. The transaction logs store all changes that need to be made to the underlying database.

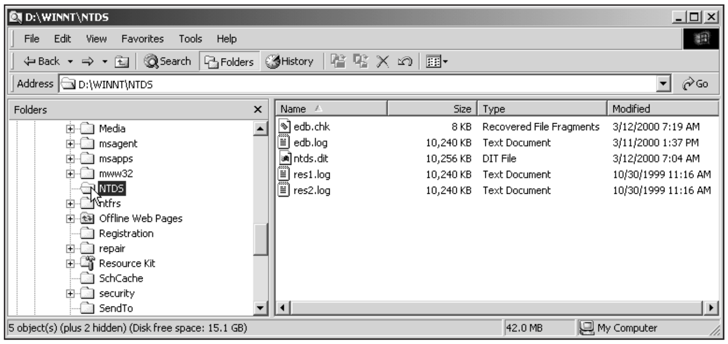
The checkpoint file EDB.CHK, on the other hand, is simply a file that contains pointers to where transactions begin and end in the transaction log file. Active Directory does not really *need* this file, because it could work out the information simply by reading the transaction logs directly; but doing so would be less efficient.

By looking at the checkpoint file, Active Directory can write any uncommitted transactions to the database. This process takes place at system startup. So, if a server is suddenly turned off and then turned back on, Active Directory reads this file to make sure the database file is up to date.

## PATCH FILES

Patch files, which have a PAT extension, are used during online backups. An **online backup** means that you can perform a backup operation even though the underlying data is still being used. Doing so introduces additional complexity that is taken care of by patch files.

For instance, what happens if a transaction is recorded during the backup process, and the underlying data in ntds.dit that will be changed by that transaction has already been backed up to tape? The transaction is written to the patch file. This file is backed up along with all the other Active Directory files. It is then deleted from the server (which means you will not get to see the patch file). A typical folder listing for the <*systemroot*>\NTDS folder, from which the patch files have been deleted, is shown in Figure 9-5.



| Name | Size | Type | Modified |
|---|---|---|---|
| edb.chk | 8 KB | Recovered File Fragments | 3/12/2000 7:19 AM |
| edb.log | 10,240 KB | Text Document | 3/11/2000 1:37 PM |
| ntds.dit | 10,256 KB | DIT File | 3/12/2000 7:04 AM |
| res1.log | 10,240 KB | Text Document | 10/30/1999 11:16 AM |
| res2.log | 10,240 KB | Text Document | 10/30/1999 11:16 AM |

5 object(s) (plus 2 hidden) (Disk free space: 15.1 GB) 42.0 MB My Computer

**Figure 9-5** Listing for the NTDS folder

## ACTIVE DIRECTORY DEPENDENCIES

It is important to understand that Active Directory does not exist in a vacuum. We have so far talked about a set of files that are an integral part of Active Directory and that must be backed up if you are to have any chance of salvaging a server that has crashed. If you have only those files, however, you will quickly find that you do not have enough information to restore a server fully.

The additional files and services that you need are collectively known as the **system state data**. This data should be backed up regularly.

### System State Data

For an Active Directory restore to be successful, you need to return the server to the state it was in prior to the condition that caused it to fail. To do so, you must restore dependent services along with the Active Directory files. In fact, this information is so important that you cannot restore Active Directory without it.

What is this system data? First, consider that if Windows 2000 will not boot, restoring the Active Directory data won't do you any good. You must be sure you back up the system startup files—without them, the system is not operational.

Second, don't forget the system Registry. The Registry in Windows 2000, much as in previous versions, is the most important file if you want to maintain the settings on a server. A corrupt or missing Registry is fatal to a system.

Some other services and components must be backed up if they're installed on a Windows 2000 DC. These include DNS, Certificate Server, and all File Replication Service settings. These elements work on your network to both support and implement various pieces of Active Directory; if they are not working, you will not be able to restore a DC to an operational state.

Note

You cannot restore a DC in Windows 2000 unless you have a backup of the system state data. Failure to back up this data will prevent you from performing a restore.

## TYPES OF RESTORE

You can perform a restore of a Windows 2000 DC using two different modes: non-authoritative and authoritative. This choice is due to the additional complexities of Active Directory and its replication model. If you think about how replication works for a moment, you can quickly identify the problems.

All DCs in a Windows 2000 domain are peers of one another. That is, changes can be made at any DC, and those changes will be replicated to all the other DCs in the domain. If a DC crashes and has to be restored, the Active Directory data that is pulled from backup media (usually tape) is by its very nature out of date. What happens when you have a fully operational DC, but its data is inaccurate?

You have two choices in such a circumstance: You can perform a restore that simply gets a DC up and running, and then let Windows 2000 take care of the rest; or you can dictate a precedence for the data. We will define these types of restore next. As well as being useful for disaster recovery purposes, knowing about them will enable you to recover from accidental mass deletions—so this information can be very useful!

### Non-Authoritative Restore

A **non-authoritative restore** is the most common method of performing a restore of a Windows 2000 DC. In this case, a restore is performed, and then Active Directory replication takes care of the rest. Because the data on the backup media will be out of date, normal replication processes make sure that all changes not currently recorded at the replica are made.

Don't forget that a restore procedure restores much more data than simply the Active Directory files. Because of this fact, the DC must be brought offline for a restore to take place.

Because you cannot be sure that the Active Directory data is consistent, its integrity must be checked before the DC is operational. To take care of this task, the restore operation writes a Registry key that causes the DC to perform both a consistency check and a re-indexing of all Active Directory data. You will not see this key, because once the operation is completed the key is deleted from the Registry.

Once these tests have taken place, the server can be brought back online. Normal Active Directory data replication processes now kick in, and the DC is able to receive all changes that were made after the backup was run.

In the case of non-authoritative restores, a DC is simply brought back online; it then accepts changes from other DCs and conforms to their view of the directory. Any changes made are recorded on the replica. This is the default mode for restores.

## Authoritative Restore

In the case of an **authoritative restore**, data that has been restored is given precedence over data stored on every other DC in the domain. This might sound like a strange thing to do—let's go through a scenario to explain why such a restore is sometimes useful.

Joe the system administrator is having a bad week. The network has been slow all day, and e-mail is down. People are running in and out of his office, complaining and asking him to hurry up, he has not had lunch, and it's four in the afternoon. About this time, his boss comes in, saying, "You have to delete that Temp group right now—they have all left the company, and it's a security problem!"

Joe is already quite flustered, and now he's also annoyed that his boss doesn't realize he is very busy. So, he pulls up the Active Directory Users and Computers administration tool, highlights what he thinks is the Temp group, and presses Delete. Then he goes back to the tasks at hand.

A little while later, he starts getting further complaints from many different departments. They no longer have access to certain network resources—and some of their desktop settings have disappeared. Joe opens his administration tool again and finds, much to his surprise, that he didn't delete the Temp group—instead, he inadvertently deleted an entire Organizational Unit (OU)! He frantically looks for the Undo function, and then realizes Windows 2000 doesn't have one. Joe breaks into a cold sweat, wondering how he's going to fix this problem.

Luckily for Joe, he performed an up-to-date backup this morning. A non-authoritative restore, as we have just seen, will do nothing to help him in this circumstance—but an authoritative restore will. An authoritative restore allows you to perform a restore and then tag certain objects to be authoritative. When they are marked as authoritative, changes to those objects will not be accepted from other DCs when the restored DC is brought online. In fact, the opposite process takes place: The flagged objects replicate to all other DCs.

Joe performs an authoritative restore for the OU he created and lets replication take care of the rest. Within an hour, his users are back at work. All the DCs have accepted the replication from the restored server, he's finally able to eat lunch, and life is good again. An example of the conversation between DCs when an authoritative restore is performed is shown in Figure 9-6.

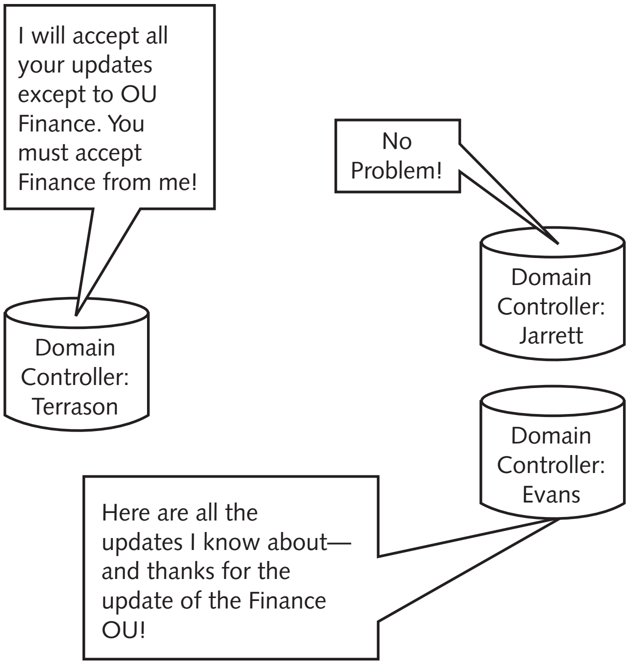

I will accept all your updates except to OU Finance. You must accept Finance from me!
No Problem!
Domain Controller: Jarrett
Domain Controller: Terrason
Domain Controller: Evans
Here are all the updates I know about—and thanks for the update of the Finance OU!

**Figure 9-6** An authoritative restore

In order to perform an authoritative restore, you must use a command-line tool called NTDSUTIL. This tool is somewhat complex, but useful.

## Strategies for Active Directory Backup

Most people are quick to realize that performing a backup of data on a DC is a good idea. After all, if something can go wrong, it will, and many of us have experienced a hard disk failure.

In order to facilitate a solid backup strategy, you should include certain things in your backup plan. The first of these is an optimal hardware configuration. Running a Windows 2000 DC on a system with a single Integrated Disk Electronics (IDE) drive will not provide you with optimal backup and restore conditions (not to mention performance).

It is also important that you come up with a comprehensive backup plan, *and then stick to it*. It is no good having all the hardware in the world and a great plan on paper, if you never execute that plan. You must commit to performing backups regularly, and yet hope you never need those backups. If you do need them, however, they mean the world.

### Hardware Configuration of Domain Controllers

When you get down to it, the bigger the box, the better. In this section, we will talk about some basic disk configuration information that will enable you to optimize your domain controllers. Although this information does not directly relate to backup and restore procedures, it is a key concern when making your recovery plans.

We all know we should have fault tolerance on our servers. Usually, that equates to RAID 5, but, although RAID 5 has a lot going for it, it does not achieve everything we would hope for in the way of performance tuning. It can certainly save you if a disk goes bad—perhaps enough to prevent you from ever having to resort to your backups—but you can do some other things that will yield better performance.

For the best performance, you must ensure both fault tolerance and separate physical disks for various files. RAID 5 supplies the former, but not the latter. Although separate disks will undoubtedly increase the hardware costs and complexity of the hardware configuration on your servers, the performance gains can be considerable.

Three sets of files should exist on their own physical disks. They are:

- Windows 2000 operating system files (\WINNT)
- Active Directory database file (ntds.dit)
- Transaction logs (EDB.LOG)

This arrangement will give you optimal performance on your servers. Even if you can achieve only part of your goal—perhaps putting the operating system files on one physical disk and the Active Directory files on another—doing so will give you superior performance over having all the files on a single set of disks. Anything you can do in this area is beneficial.

## ACTIVE DIRECTORY MAINTENANCE

Now that we have looked at backing up and restoring Active Directory, let's turn to the topic of how to maintain it. As you have seen, once you strip out all the magic that Active Directory can perform, you are left with a database and a set of files. Like most files on a hard disk, some periodic maintenance is required to keep them working at peak efficiency.

Maintenance is important for several reasons. As objects are deleted from Active Directory, space is not necessarily reused; and as with many databases, this can cause fragmentation. A database that takes up more space on the physical disk than necessary is obviously a problem, and a fragmented database is inefficient and causes processing overhead.

Although some of these issues are dealt with automatically on a regular schedule, we will show how you can perform them manually. This information will give you the flexibility to decide when these processes take place.

## Automatic Maintenance

Fortunately, Microsoft has taken care of two of the most pressing issues that can occur on a Windows 2000 DC: fragmentation and the deletion of objects. This process has a rather stern name—it's called *garbage collection*. The garbage collection process kicks off on each DC every 12 hours.

This process performs three key tasks, which are outlined in the following sections. (Don't forget that you can also perform these tasks manually, as we will discuss a little later.)

### Deleting Transaction Logs

As you saw earlier, the transaction log (EDB.LOG) can be 10 MB in size by default. When the log file gets full, it is renamed, and a new EDB.LOG file is created. After all transactions in a log file have been written to the database, the log is no longer required. Garbage collection identifies these log files and deletes them.

### Deleting Objects from the Database

The dynamic nature of Active Directory replication introduces a few complications into database maintenance. Let's take object deletion as an example.

When changes are made to Active Directory (such as deletion of a user), this information originates at one DC and then must be replicated to all other DCs. Before the object can be physically deleted from the database on each DC, Active Directory must be sure that all replicas are aware of the deletion and are in fact going to make the deletion, as well. This process ensures consistency of Active Directory data. Without it, one Active Directory database might think an object exists when in fact it has been deleted from every other replica.

To deal with such situations, Microsoft has implemented a system known as a **tombstone lifetime**. When an object is deleted from Active Directory, it is flagged with a tombstone lifetime; the default value is 60 days. This period gives Active Directory time to copy the deletion request to all replicas. In the meantime, the deleted object does not show up in user queries, giving the impression that it is gone.

Note

Tombstoning is a key element of Active Directory design because it involves the permanent deletion of data from the directory. Once the data is gone, it is permanently removed—you can't get it back. The date for the deletion will be the same on each DC, so when the data is removed, it no longer exists anywhere on the network.

You might think that you could get the data back, even if the tombstone lifetime has expired and the data has been deleted, simply by performing a restore. You'd be wrong. Because all references—on all DCs—are removed, it is not possible to use a tape that is older than the tombstone lifetime setting in Windows 2000. You should take into account the default time period of 60 days when making your backup plans. You cannot restore data from tape once the data's tombstone lifetime has expired. If doing so were allowed, replication would fail.

When garbage collection is taking place, each object in the directory is checked to see if its tombstone lifetime has expired. If it has, the object is deleted from Active Directory.

### Database Defragmentation

As objects are deleted from Active Directory, fragmentation can occur. For example, suppose a 4K object is deleted from the database, and then an 8K object is created. Active Directory, like most file systems, will copy half of that new object's data to the 4K space and the remaining data to a different location within the file. This process should not be considered a problem, because all current file systems work this way.

The automatic cleanup process performs a defragmentation of the database. Doing so ensures optimal performance of database queries. Because the cleanup process operates every 12 hours, the database is kept finely tuned.

Note

It is important to note that performing a defragmentation does not decrease the size of the database. To decrease the physical size of the database, you must run the manual cleanup process. The database can require resizing when a large number of objects have been deleted.

## Manual Database Cleanup

By now you should be convinced that maintenance of the Active Directory database is a good idea. You have seen how a process kicks off on each domain controller every 12 hours and performs most of the necessary tasks to ensure that things run smoothly and that your servers perform to the highest degree of efficiency.

There are a couple of tasks that the automatic process does not perform, however. These tasks should be taken care of in a manual process. The key points to know are:

- Manual cleanup reduces the size of the database; automatic cleanup cannot.
- Manual cleanup involves taking the DC offline.

We will briefly discuss these points in the section that follows. It is important to note that although some automatic processes take place, periodically you will have to schedule a manual maintenance process to ensure that your directory is performing at its best.

### Offline Defragmentation

Unlike the automatic cleanup process (online defragmentation), the offline version of this process can actually reduce the size of an Active Directory database file. Microsoft has stated that you should never need to run an offline defragmentation; however, if there have been many deletions from your directory, it is the only way to reduce the physical database file to a more reasonable size.

We highly recommend that you perform a test of this process in a lab before you take a production DC offline. Doing so will enable you to make sure that the process works on your system, and to see the benefits of running it. These benefits will vary depending on how your directory is being used.

You should not be concerned that a DC that is taken offline will no longer be up to date when it is turned back on. The Active Directory replication process will ensure that any changes made after the DC was shut down will be replicated.

## Basic Backup Principles

Effective backup is accomplished through two tasks. First, you must have an understanding of the underlying technical issues, have a clear idea of *what* should be backed up, and understand the situations that might prevent you from performing a restore operation. These concepts were covered in the first part of this chapter.

Second, you should keep in mind some basic but important points that must be addressed in order to ensure that no matter what happens, a restore operation will be a viable option. You can do some simple things to make sure everything runs smoothly, as we will explain in the following sections.

### Hardware Requirements

Generally, the media used for backup is tape. Many different manufacturers make tape drives and tape media, and we will not try to rate them. However, you need to be aware of several considerations:

- Make sure you have sufficient capacity on your tape drive. If you have 10GB of data to back up, and your tape drive has a capacity for only 2GB, then you will need five tapes in order to perform a backup. This need involves far greater interaction from the system administrator (and wastes the administrator's valuable time). It will also increase the amount of time a backup takes.
- If the backup process is slow and cumbersome, it may never get done. The number one problem with backups is that often they are not performed, because the process is time consuming and difficult.
- Be wary of hardware vendors' claims about the capacity of their tape drives. Often, a drive that claims to store 20GB of data is actually a 10GB-capacity drive with compression turned on. Compression is an effective way of getting more data into a smaller space, but, like much advertising, the reality rarely meets the expectations set by marketing and sales departments. Assume the worst when buying hardware—you can never have too much capacity.
- Don't forget that performing a backup causes a lot of network traffic. On most corporate networks, you won't want this process to occur in the middle of the afternoon. Schedule the backup to occur when a minimal number of users are on the network. For many organizations, this is some time in the early hours of the morning.

### Media Storage

You need two sets of backup media. You must have a set close at hand, so when someone asks you to restore a file or DC, you can do so quickly. However, a second set of media should be stored at an off-site location, to protect you from acts of God, such as fire and flood. If your entire office burned to the ground, what would you lose? When it comes to data, the answer should be "nothing." (You have probably heard this warning many times, but are you actually heeding it?)

It does not matter where the media are taken, as long as they are not in the same physical building as your other media. Everyone should know the storage location, so they can retrieve media if needed. Don't forget that data is the lifeblood of a business; so, the storage area must not only be distant from the location of the servers, but it also should be both secure and media friendly. **Media friendly** means a controlled environment. There is little point in throwing the media into a box at the back of your garage, if the temperature there reaches 110 degrees every day.

Finally, don't forget that the tombstone lifetime is set by default to 60 days. Tapes containing Active Directory are no longer much use after that time.

### Testing Restore Operations

Do not assume that the shiny new tape drive and media, along with perfect error logs from your backup software, are doing everything they say they are doing. You should plan to perform test restores periodically, to make sure that data is retrievable. If you do not, there is a chance that your backup will fail when you need it most—after a system failure.

You should also have redundant backup hardware. The media backed up with one device should be used for restore purposes in the opposite hardware. This process ensures not only that the media is good, but also that the hardware is working properly. Although this setup obviously increases the costs involved with performing backup operations, there is no substitute when the chips are down and the network is on its knees due to a server crash.

## Backup and Restore Security

The data on your network is critical, not only because it needs to be available at all times, but also because it must be protected from unauthorized access. The only real way to make a server secure is to provide for physical security. That is, the server must be in a locked room, and only a small number of people should have physical access to it.

Of course, this arrangement becomes somewhat complicated when you are talking about backups. The media you are using must be secured—particularly the off-site media. Always make sure there is accountability for this media.

It would also not make much sense if anyone could perform a backup and restore of Active Directory data. Toward this end, Microsoft has defined two sets of built-in groups for this purpose: one for performing backups and one for performing restores. As a result, you can assign these two important tasks to different groups of people.

The only people who can perform backup and restore operations are members of the Domain Administrators and Backup Operators groups. Alternatively, you can create groups yourself. For this to work, you must assign the groups both the Backup File And permissions and Restore Files And Directories permissions.

To perform a restore when a DC is offline, you must be a member of the Local Administrators group. This requirement runs somewhat contrary to previous Microsoft Windows NT behavior. In versions prior to Windows 2000, DCs had no local account. You will need to know this account name and password to perform a restore.

## THE MICROSOFT BACKUP UTILITY

Microsoft has been kind enough to provide a Backup utility with Windows 2000. Do not confuse this with a full-fledged enterprise-level backup program, however. The utility provided by Microsoft lacks some significant features, and you should be looking for a more fully featured application to use on your own production servers. For example, not least of the missing features is the utility's inability to back up data on remote servers; the backup utility can back up data only on the local machine.

The Backup utility provided is actually a scaled-down version of Backup Exec, a product from Veritas Software (www.veritas.com). Despite its lack of enterprise-level features, this utility can perform all of the necessary functions mentioned in this chapter.

## CHAPTER SUMMARY

- In this chapter, we examined both recovery and maintenance of Active Directory. We looked at how you can protect your network from catastrophic loss of a domain controller in Windows 2000. In order to fully recover from a crash, you should have a full backup of your data.
- To help you fully understand how Active Directory operates, we first took a look at the architectural model. This model enabled us to show how a DC is able to guarantee consistency of data and allow for recovery from simple failures such as a power loss. More specifically, we examined the three-layer model, including the Directory System Agent (DSA), the database layer, and the Extensible Storage Engine.
- These three layers have distinct roles, and understanding each of these roles enables you to gain a better understanding of how data is stored in Active Directory (and why some files are more important than others for backup purposes). The DSA is responsible for retrieving data for client applications. Clients never directly access the data store of Active Directory.

- The database layer is simply a view of the data. The actual data is stored in a flat file; data in Active Directory is often viewed in a hierarchical fashion, but the physical storage is simply a table with rows and columns, much like a spreadsheet. The database layer organizes the data into a hierarchical form, so it is easier to read and understand.
- The ESE is the underlying data engine that powers Active Directory. We learned that this engine has been tested extensively in the real world—because it is the same engine used in Exchange 5.5.
- In order to understand how Windows 2000 maintains resiliency on DCs, we took a close look at how data is written to the database. This process involves interaction among several different files. The process might appear to be overly complex, but it is designed to ensure that DCs can recover, should there be a simple failure such as power outage.
- We learned that data is not written directly to the Active Directory database. Instead, data is first written to a transaction log. Once the entire transaction is written to the log, it is written to a piece of the database that is buffered in memory. Finally, it is written to the database on the disk. If any one of these steps fails, then the transaction is rolled back, which means it is undone at every level.
- In order to understand which files you should be certain to back up, we took a look at the physical files that support Active Directory and discussed which files would be important, should you need to perform a restore operation. We began with ntds.dit, which contains all of the Active Directory data. We learned that two copies of this file exist: one is used for installation purposes only and is stored in the <*systemroot*>\system32 folder, and the other is the actual Active Directory data file.
- Next, we examined the various log files used by Active Directory. Only one of these files, EDB.LOG, is used all the time. This file is a transaction log, which means all transactions are first written to this file before being applied to ntds.dit. A transaction log is useful because a Windows 2000 DC can quickly read this file to find out if it missed any transactions. This process occurs during each system boot-up. The default size for this log file is 10MB.
- There are two other log files: RES1.LOG and RES2.LOG. The main purpose of these two files is to reserve space on a DC's hard disk, in case the disk drive that stores Active Directory data becomes full and transactions can no longer be written to EDB.LOG. In this case, transactions are written to these two placeholders. Each of these files is 10MB in size.
- Logging is an important task for any DC. We learned that when a log file reaches its maximum size (10MB), a copy of the log file is made, and a new one is created. This behavior affords maximum recoverability; however, it requires additional disk space (10MB multiplied by the number of backup files created).

If you do not have the necessary resources, you can turn on circular logging. With circular logging, backup copies of old transaction log files are not made.

- As data is written from the transaction log to the underlying database, the transaction is considered **committed**. A pointer is written to indicate the last transaction that has been committed in the log file. This information is in turn stored in a checkpoint file named EDB.CHK.
- We also took a brief look at patch files. These files are used during online backups. An online backup takes place while the DC is being used for its normal, everyday purposes. Patch files ensure that data changed during the backup operation is recorded on the backup media. Without these files, you would find inconsistencies in the data, should it be used for a restore.
- Active Directory does not work alone. We learned that fully restoring a Windows 2000 DC also requires a full copy of the system state data. This data includes everything needed to bring a server back to the same state it was in before the failure, including operating system files, the Registry, and any services installed on the DC (such as DNS services and the File Replication Service). These services are integral to Windows 2000, and failure to back them up will cause you to be unable to achieve a completely operational state even if you have backed up the Active Directory files.
- You can use two types of restore operation: non-authoritative, which is the default operation; and authoritative. With a non-authoritative restore, once a restore operation is complete, normal Active Directory replication takes place to bring the DC up to date. With an authoritative restore, you are able to flag certain objects to replicate to every DC within the domain. In essence, the restored data becomes the master data, and each DC will accept it. This operation is useful when you have accidentally deleted an object, and you need to get it back.
- Backup is only one of several ways you can protect data on your DC. We briefly discussed other methods, such as using RAID 5. For optimal configurations, you should try to ensure that the three sets of essential files—ntds.dit, EDB.LOG, and the operating system files—are stored on different physical disks.
- We discovered that several maintenance tasks take place periodically on a DC. They occur without user interaction every 12 hours by default. This maintenance procedure defragments the NTDS.DIT file, deletes any objects from the database, and removes transaction log files that are no longer needed.
- Objects are physically deleted from the Active Directory database only after a tombstone lifetime flag has expired. This flag is set to 60 days by default. It is set because Windows 2000 needs to ensure that all DCs are aware that the object should be deleted. If this did not happen, there would be inconsistencies in Active Directory data. Tombstoning has a significant effect on system restores, because media older than the tombstone life (60 days) cannot be used.

- We learned that although this maintenance process takes place automatically, you might sometimes want to perform it manually. Performing manual maintenance also enables you to shrink the size of the Active Directory database. For manual maintenance tasks, however, the DC must be taken offline (the automatic maintenance process does not need to take the server offline).
- Finally, we examined a few important considerations to keep in mind when designing your backup strategy. You should ensure that you have sufficient hardware to perform the necessary backups. We also discussed keeping a copy of backup media off-site, to protect you in case of a catastrophic failure in which you lose your storage area at the local office. We also suggested that you perform periodic test restores to make sure that restore operations are possible, and to ensure that the backup hardware, the backup software, and the physical backup media are all functioning properly.